Spring Framework Petclinic

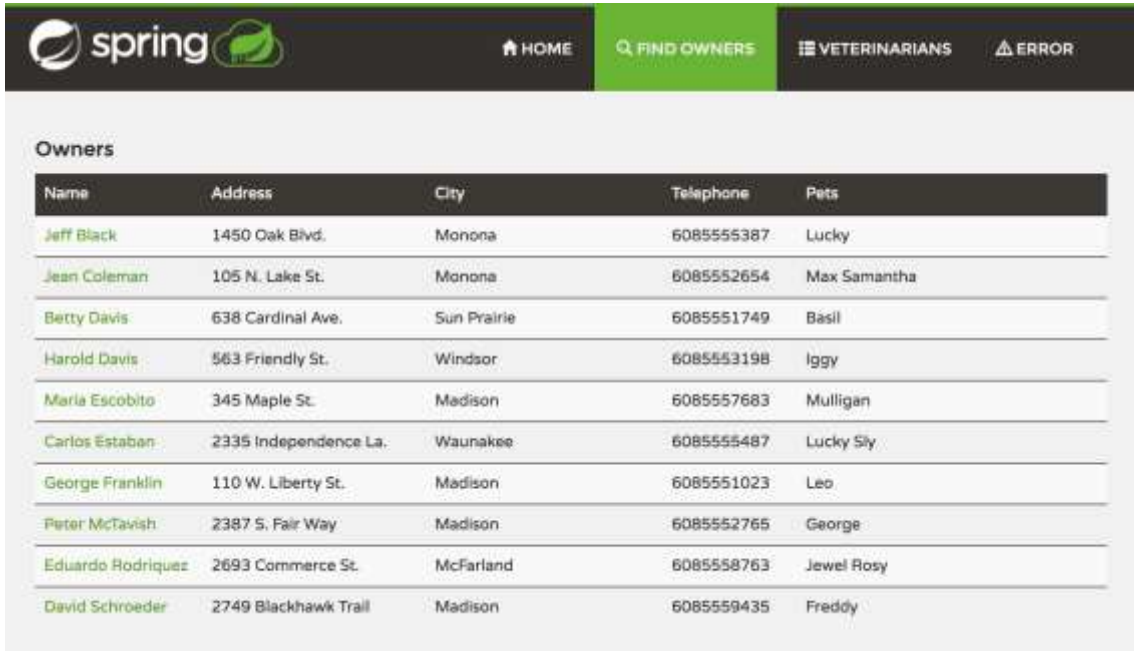
Michael Isvy
Antoine Rey

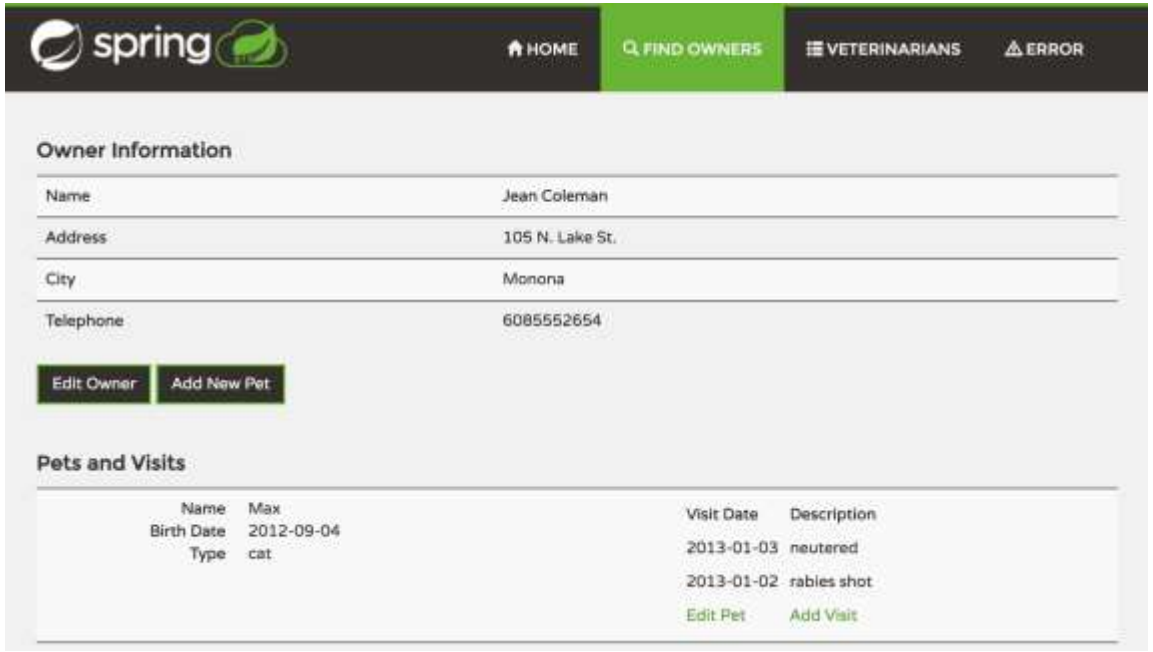
Spring Petclinic

- Sample application designed to show how the Spring application frameworks can be used to build simple, but powerful database-oriented applications
- Demonstrate the use of Spring's core functionality:
  - JavaBeans based application configuration using Inversion of Control (IoC)
  - Model View Controller (MVC) web Presentation Layer
  - Practical database access through JDBC, Java Persistence API (JPA) or Spring Data JPA
  - Application monitoring based on JMX
  - Declarative Transaction Management using AOP
  - Data Validation that supports but is not dependent on the Presentation Layer
- Exists many versions (forks) of the Spring Petclinic sample application

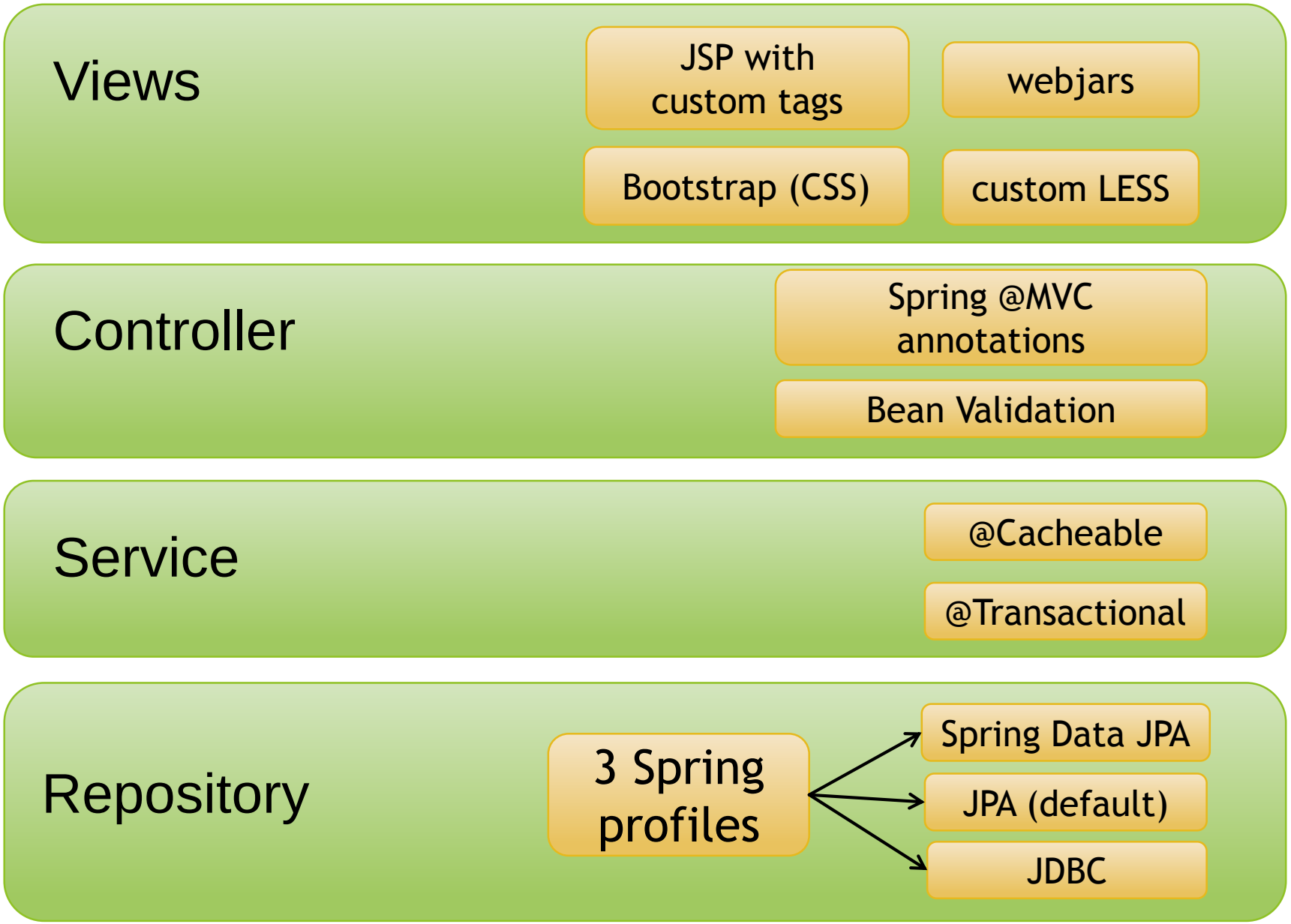
Spring Framework Petclinic

- https://github.com/spring-petclinic/spring-framework-petclinic
- **Fork** of the « canonical » implementation of Spring Petclinic
- Maintain a Petclinic version with a plain old **Spring Framework configuration** and with a **3-layer architecture**

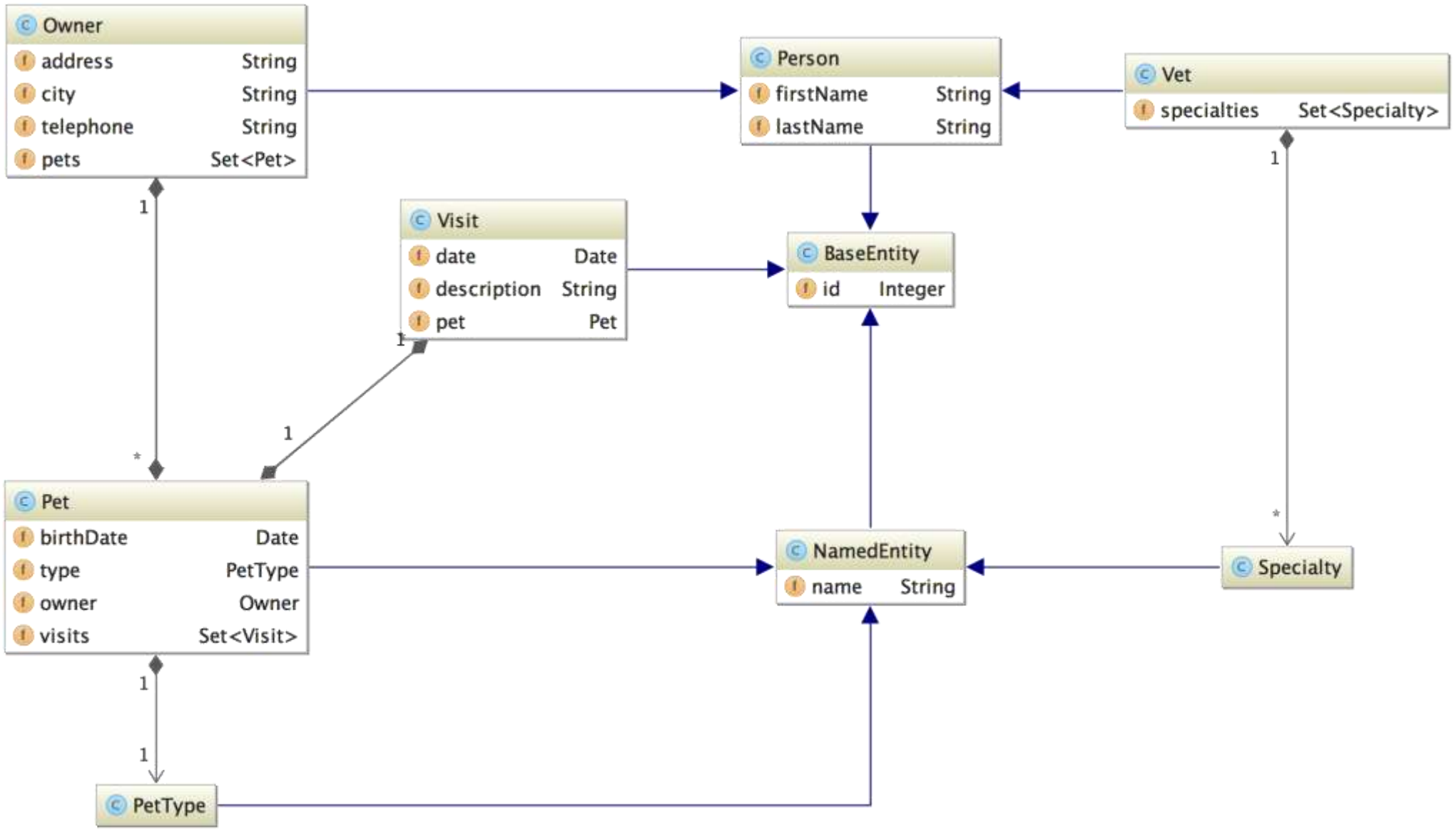




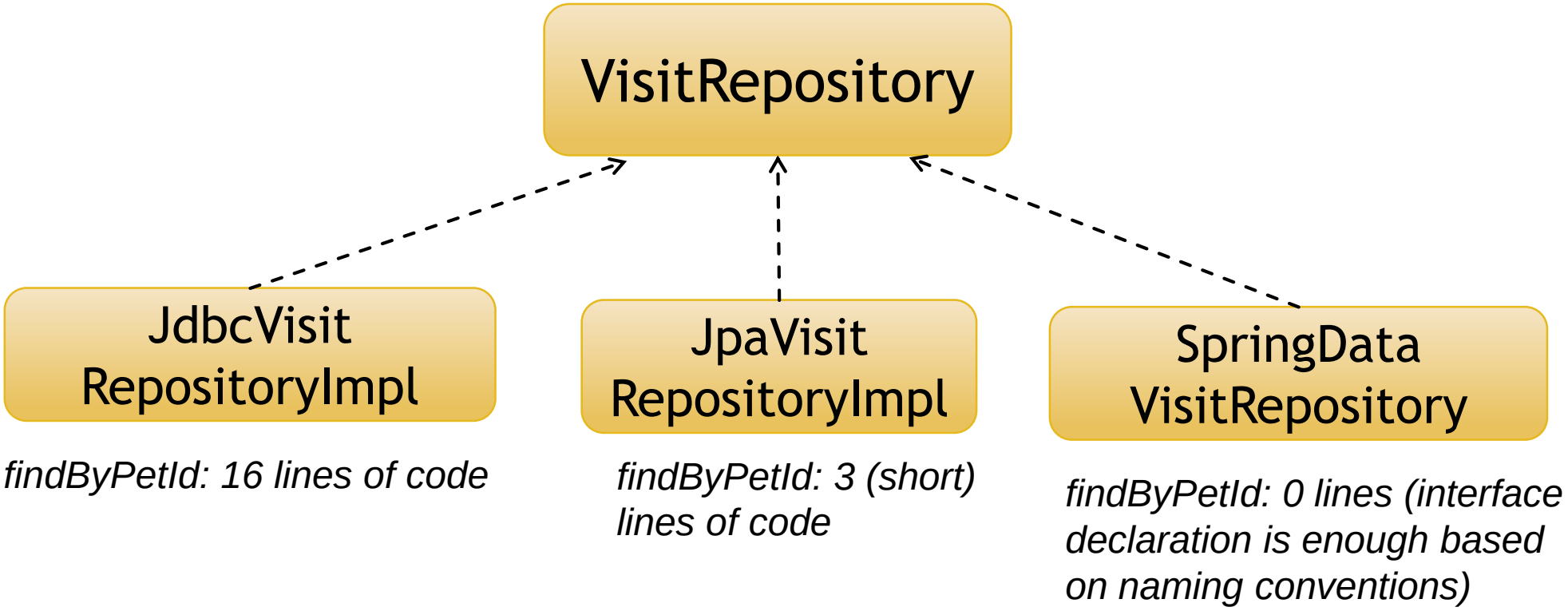
Software Layers



Domain Model



Data Access



In order to select which implementation should be used :

1. select the appropriate bean profile inside PetclinicInitializer (jdbc, jpa or spring-data-jpa)
2. or use the *-Dspring.profiles.active=jdbc* VM option

Database

- Supports HSQLDB (default), MySQL, PostgreSQL
- Connections parameters and drivers are declared into Maven profiles
- DDL and DML SQL scripts for each database vendors:

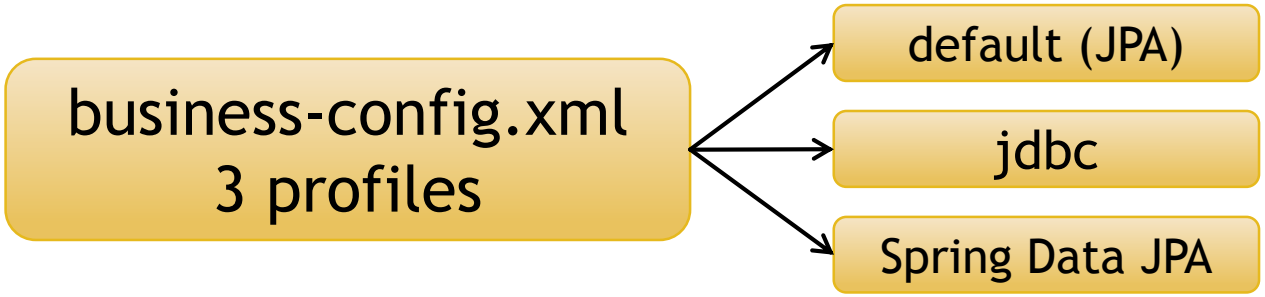
```
# Properties that control the population of schema and data for a new data source
jdbc.initLocation=classpath:db/${db.script}/initDB.sql
jdbc.dataLocation=classpath:db/${db.script}/populateDB.sql
```

data-access.properties

- How to start Spring Petclinic with a MySQL database?

```
docker run --name mysql-petclinic -e MYSQL_ROOT_PASSWORD=petclinic -e MYSQL_DATABASE=petclinic
                                  -p 3306:3306 mysql:5.7.8
mvn tomcat7:run-war -P MySQL
```

Bean profiles




```
<beans profile="jpa,spring-data-jpa">
   <bean id="entityManagerFactory" … >
</beans>
```

Inside Junit tests

```
@ContextConfiguration(locations = … })
@RunWith(SpringJUnit4ClassRunner.class)
@ActiveProfiles("jpa")
public class ClinicServiceJpaTests … { }
```

Inside PetclinicInitializer.java

```
XmlWebApplicationContext context =
            new XmlWebApplicationContext();
context.setConfigLocations(…);
context.getEnvironment().setDefaultProfiles("jpa");
```

No configuration needed in case you wish to use the default profile (jpa)

Caching

- The list of Veterinarians is cached using ehcache

```
@Cacheable(value = "vets")
public Collection<Vet> findVets()
        throws DataAccessException {...}
```

ClinicServiceImpl

```
<cache name="vets"
      timeToLiveSeconds="60"
      maxElementsInMemory="100" .../>
```

ehcache.xml

```
<!-- Enables scanning for @Cacheable annotation -->
<cache:annotation-driven/>

<bean id="cacheManager"
class="org.springframework.cache.ehcache.EhCacheCacheManager"
   p:cacheManager-ref="ehcache"/>

<bean id="ehcache"
class="org.springframework.cache.ehcache.EhCacheManagerFactoryBean"
   p:configLocation="classpath:cache/ehcache.xml"/>
```

tools-config.xml

Transaction management

business-config.xml

```
<!-- Enables scanning for @Transactional annotations -->
<tx:annotation-driven/>

<bean id="transactionManager"
class="org.springframework.orm.jpa.JpaTransactionManager"
    p:entityManagerFactory-ref="entityManagerFactory"/>

<bean id="transactionManager"
class="org.springframework.jdbc.datasource.DataSourceTransactionManager"
    p:dataSource-ref="dataSource"/>
```

Alternative to JPA, Transaction Managers for a single:

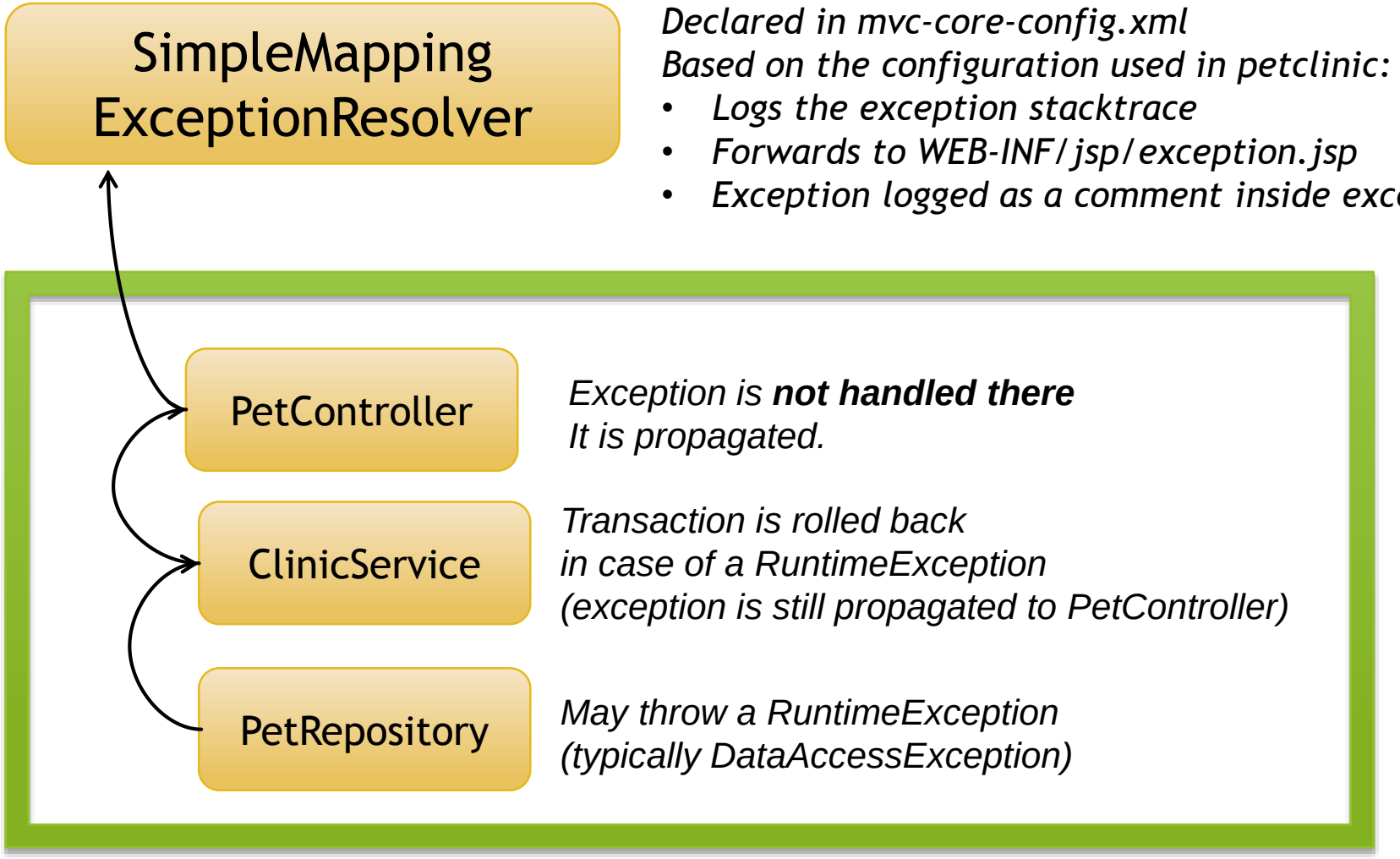
JPA *EntityManagerFactory*

JDBC DataSource

ClinicServiceImpl.java

```
@Transactional(readOnly = true)
public Collection<PetType> findPetTypes() throws DataAccessException {
   return petRepository.findPetTypes();
}
```

Exception Handling



Declared in mvc-core-config.xml
Based on the configuration used in petclinic:

- *Logs the exception stacktrace*
- *Forwards to WEB-INF/jsp/exception.jsp*
- *Exception logged as a comment inside exception.jsp*

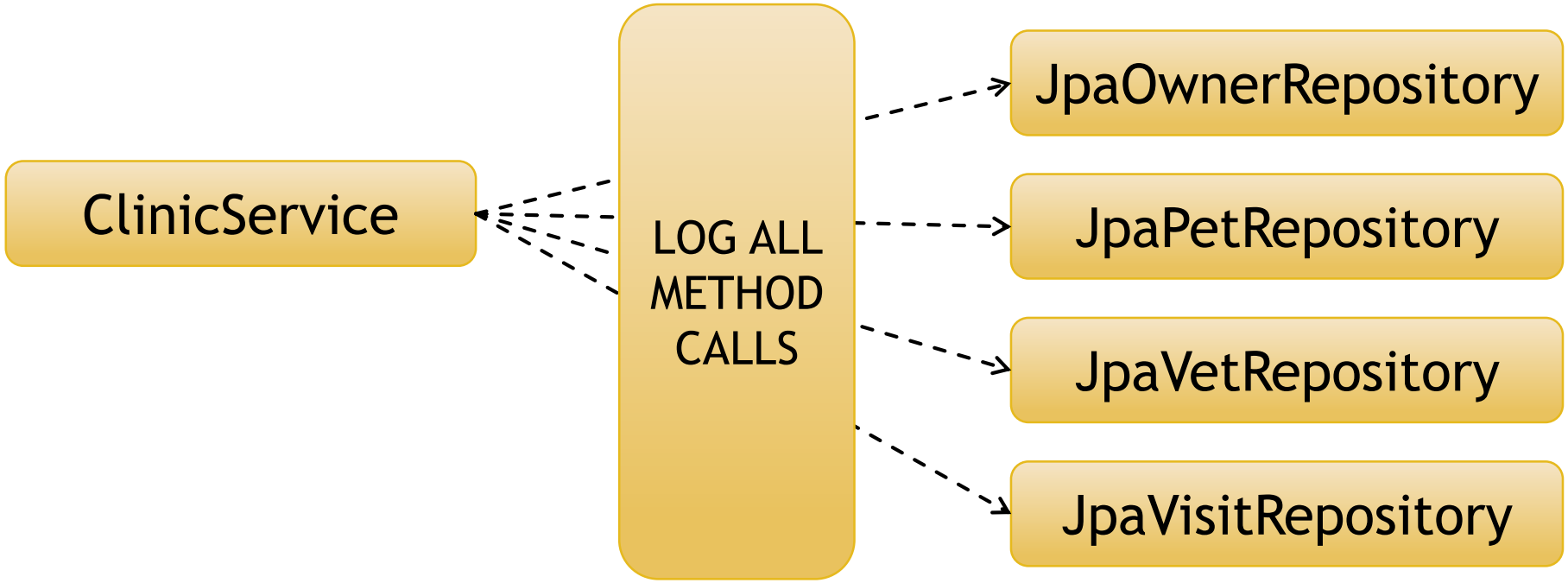
*Exception is **not handled there***
It is propagated.

Transaction is rolled back
in case of a RuntimeException
(exception is still propagated to PetController)

May throw a RuntimeException
(typically DataAccessException)

Aspect Oriented Programming (1/2)

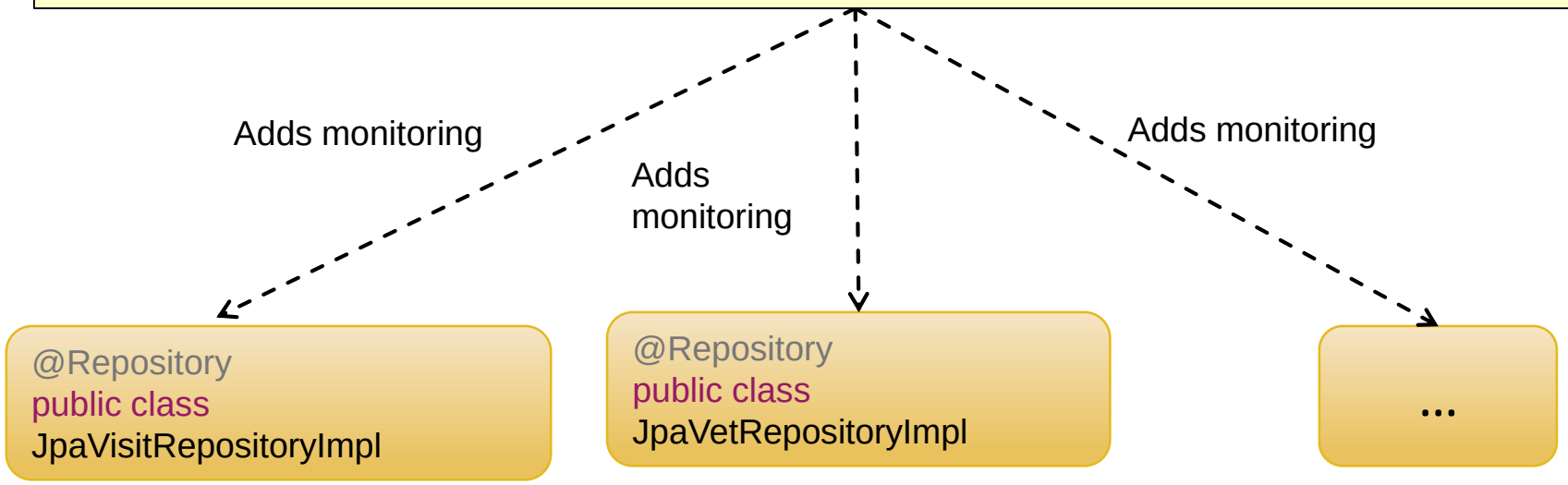
- How to add behavior in all methods of all Repository classes?



Aspect Oriented Programming (2/2)

- CallMonitoringAspect

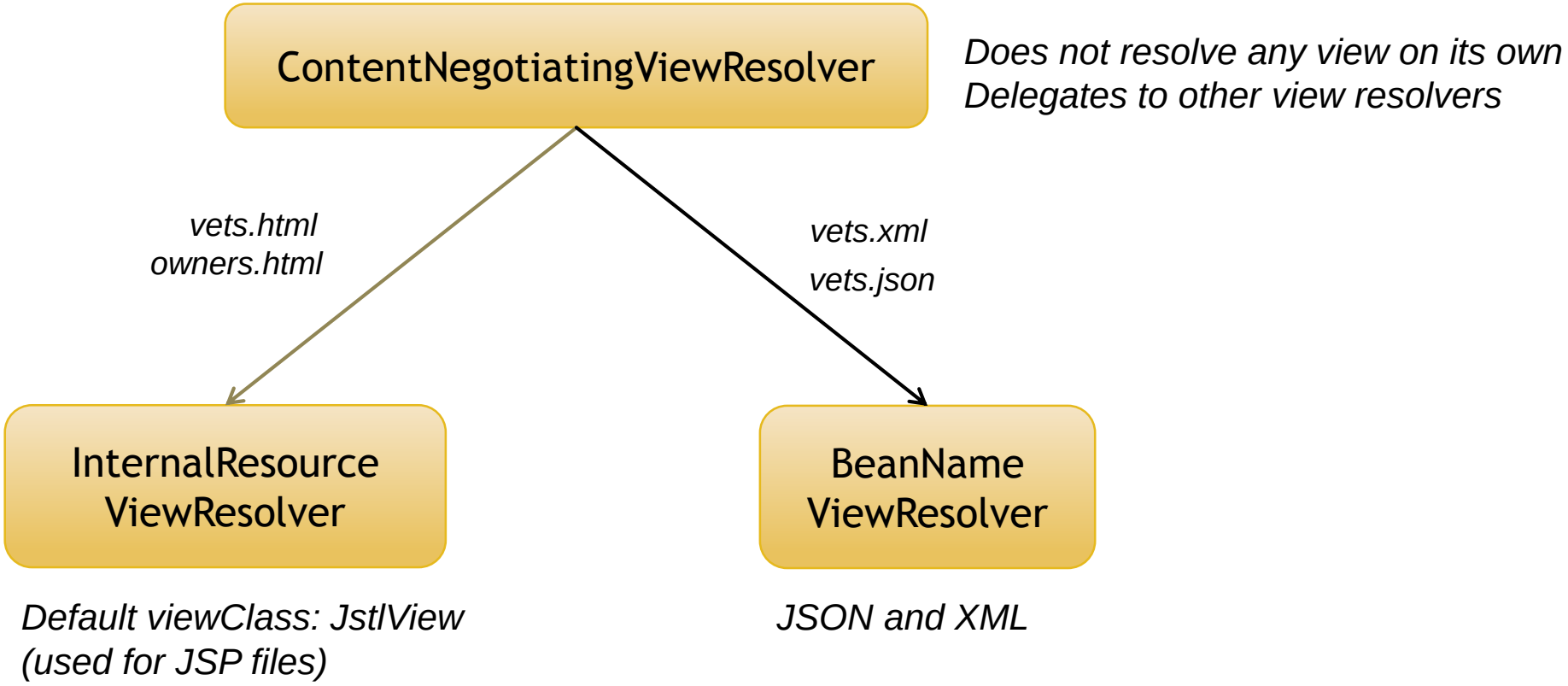
```
@Aspect
public class CallMonitoringAspect {
    @Around("within(@org.springframework.stereotype.Repository *)")
    public Object invoke(ProceedingJoinPoint joinPoint) throws Throwable { … }
```



To understand further how AOP works in Spring:
https://spring.io/blog/2012/05/23/transactions-caching-and-aop-understanding-proxy-usage-in-spring

View Resolvers in Spring Petclinic

mvc-view-config.xml



Datatables in Spring MVC

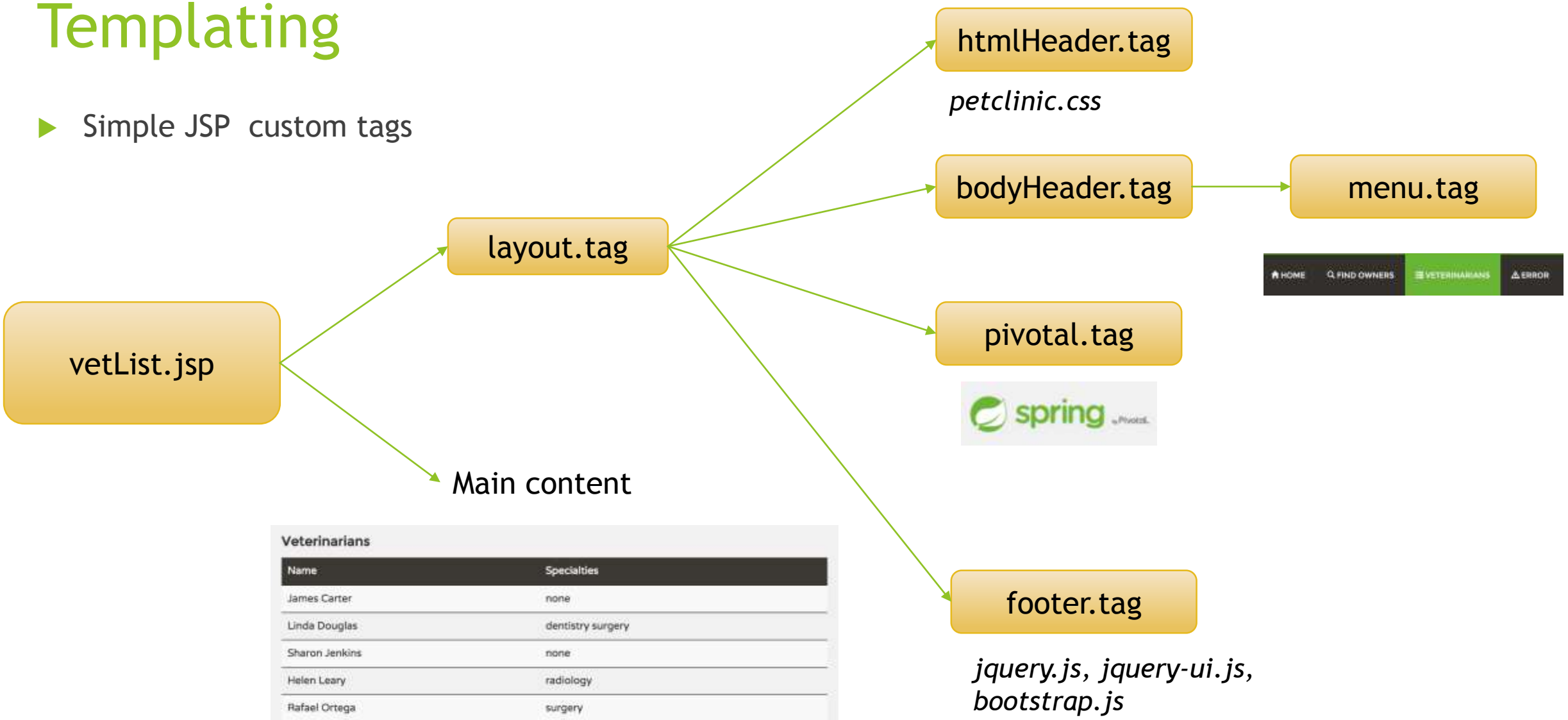
```
<table id="vets" class="table table-striped">
    <thead>
    <tr>
        <th>Name</th><th>Address</th><th>City</th><th>Telephone</th><th>Pets</th>
    </tr>
    </thead>
    <tbody>
    <c:forEach items="${selections}" var="owner">
        <tr>
            <td>
                <spring:url value="/owners/{ownerId}.html" var="ownerUrl">
                    <spring:param name="ownerId" value="${owner.id}"/>
                </spring:url>
                <a href="${fn:escapeXml(ownerUrl)}"><c:out value="${owner.firstName} ${owner.lastName}"/></a>
            </td>
            <td>
                <c:out value="${owner.address}"/>
            </td>
            …
        </tr>
    </c:forEach>
    </tbody>
</table>
```

JSP file

Simple HTML tables with Bootstrap style

Templating

- Simple JSP custom tags



Veterinarians

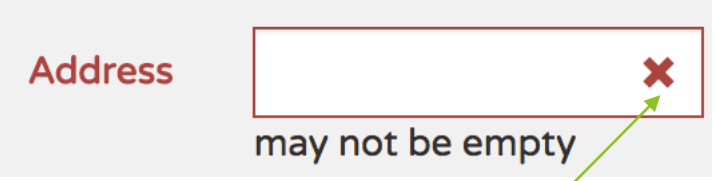
Name	Specialties
James Carter	none
Linda Douglas	dentistry surgery
Sharon Jenkins	none
Helen Leary	radiology
Rafael Ortega	surgery
Henry Stevens	radiology

View as XMLView as JSON

Validation

- Server-side validation with Bean Validation
  - Few annotations on entities: @Digits, @NotEmpty (Hibernate Validator)
- Custom Spring MVC Validator when required (i.e. PetValidator)

```
@RequestMapping(value = "/owners/new",
                        method = RequestMethod.POST)
public String processCreationForm(@Valid Owner owner,
                                  BindingResult result) {
   if (result.hasErrors()) { ...
```




```
public class Owner extends Person {
   @Column(name = "address")
   @NotEmpty
   private String address;
   ...
```

```
<c:if test="${status.error}">
   <span class="glyphicon glyphicon-remove
 form-control-feedback" aria-hidden="true"/>
   <span class="help-inline">${status.errorMessage}</span>
</c:if>
```

Webjars

- Allow CSS and JS libraries to be imported as Maven libraries
  - Used in Petclinic for jQuery, jQuery-ui, Bootstrap
  - http://www.webjars.org/

www.webjars.org/classic

WebJars | Documentation | NPM WebJars | Bower WebJars | Classic WebJars | Sponsored by BinTray & Heroku

List of Classic WebJars

jquery

Contribute a Classic WebJar | Request a Classic WebJar

Name	Versions	Build Tool: SBT / Play 2 Maven Ivy Grape Gradle Buildr Leiningen	Files
jquery	3.1.1-1	<dependency> <groupId>org.webjars</groupId> <artifactId>jquery</artifactId> <version>3.1.1-1</version> </dependency>	4 Files
JQuery Address	1.6	<dependency> <groupId>org.webjars</groupId> <artifactId>cowwoc.jquery.address</artifactId> <version>1.6</version> </dependency>	2 Files
JQuery Address	1.6	<dependency> <groupId>org.webjars</groupId> <artifactId>cowwoc-jquery-address</artifactId> <version>1.6</version> </dependency>	2 Files

Using Webjars

- Inside pom.xml

```
<dependency>
    <groupId>org.webjars</groupId>
    <artifactId>jquery</artifactId>
    <version>2.2.4</version>
</dependency>
```

- Spring MVC configuration

```
<mvc:resources mapping="/webjars/**"
        location="classpath:/META-INF/resources/webjars/"/>
```

- Inside IDE

- Inside JSP (footer.tag)

```
<spring:url value="/webjars/jquery/2.2.4/jquery.min.js" var="jQuery"/>
<script src="${jQuery}"></script>
```

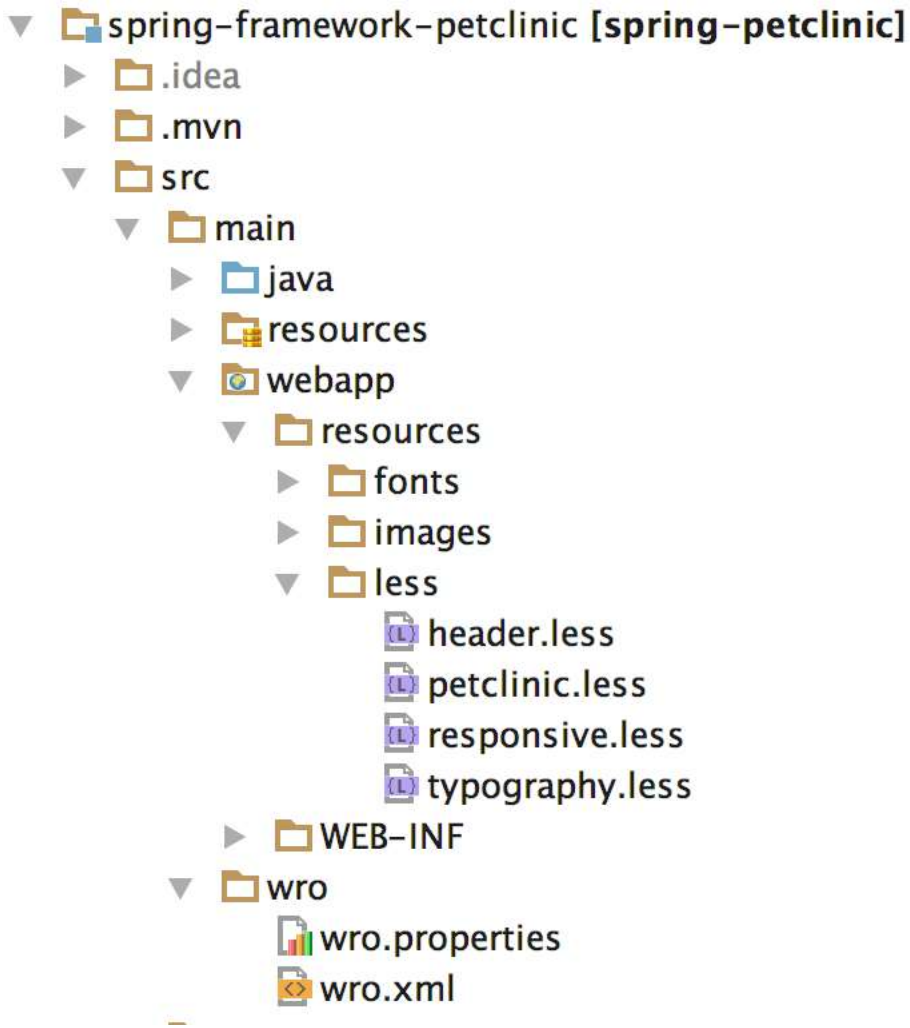
The Js file is inside a jar file!

LESS

- LESS as a CSS pre-processor
  - See petclinic.less
- CSS generated by **wro4j**
- Integrated to the **Maven** build
  - See usage of the **wro4j-maven-plugin** inside the pom.xml
- Less import from Bootstrap webjar

```
<groups xmlns="http://www.isdc.ro/wro">
  <group name="petclinic">
    <css>classpath:META-INF/resources/webjars/
          bootstrap/3.3.6/less/bootstrap.less</css>
    <css>/petclinic.less</css>
   </group>
</groups>
```

wro.xml



Java based configuration

- Spring XML configuration could be replaced by Java configuration
- Checkout the **javaconfig** branch

```
<import resource="mvc-view-config.xml"/>

<context:component-scan
   base-package="org.springfrk.samples.petclinic.web"/>

<mvc:annotation-driven />

<mvc:view-controller path="/" view-name="welcome"/>

…
```

mvc-core-config.xml

```
@Configuration
@EnableWebMvc
@Import(MvcViewConfig.class)
@ComponentScan(basePackages =
            { "org.springfrk.samples.petclinic.web" })
public class MvcCoreConfig
        extends WebMvcConfigurerAdapter {

   @Override
   public void addViewControllers(ViewControllerRegistry reg) {
      reg.addViewController("/").setViewName("welcome");
   }
   …
}
```

MvcCoreConfig.java

Unit Testing

- Frameworks: Spring Test, JUnit, HSQLDB, Mockito, AssertJ, Hamcrest, Json Path
- Tests are shared between persistence technologies
  - Inherits from AbstractClinicServiceTests

Coverage All in spring-petclinic

88% classes, 84% lines covered in package 'petclinic'

Element	Line, %
model	78% (85/108)
repository	96% (173/180)
service	100% (18/18)
util	13% (4/30)
web	95% (110/115)
PetclinicInitializer	0% (0/11)

```
@Test
public void testProcessUpdateFormHasErrors() throws Exception {
    mockMvc.perform(post("/owners/{ownerId}/pets/{petId}/edit", 1, 1)
        .param("name", "Betty")
        .param("birthDate", "2015/02/12"))
        .andExpect(model().attributeHasNoErrors("owner"))
        .andExpect(model().attributeHasErrors("pet"))
        .andExpect(status().isOk())
        .andExpect(view().name("pets/createOrUpdatePetForm"));
}
```

PetControllerTests.java

Comparing with the original Spring Petclinic

	Spring Framework Petclinic	« Canonical » Spring Petclinic
Spring stack	Plain Old Spring Framework	Spring Boot
Architecture	3 layers	Aggregate-oriented domain
Persistence	JDBC, JPA, Spring Data JPA	Spring Data JPA
View	JSP	Thymeleaf
Databases support	HSQLDB, MySQL, PostgreSQL	HSQLDB, MySQL
Containers support	Tomcat 7 and 8, Jetty 9	Embbeded Tomcat and Jetty
Java support	Java 7 and 8	Java 8

- « Canonical » implementation : https://github.com/spring-projects/spring-petclinic
- Spring Framework version : https://github.com/spring-petclinic/spring-framework-petclinic

Other Spring Petclinic versions

Name	Technologies	Github
Spring Petclinic Angular	AngularJS 1.x, Spring Boot and Spring Data JPA	https://github.com/spring-petclinic/spring-petclinic-angular1
Spring Petclinic React	ReactJS (with TypeScript) and Spring Boot	https://github.com/spring-petclinic/spring-petclinic-reactjs
Spring Petclinic Microservices	Distributed version of Spring Petclinic built with Spring Cloud	https://github.com/spring-petclinic/spring-petclinic-microservices

References

- Transactions, Caching and AOP: understanding proxy usage in Spring (Michael Isvy)
- Series of 5 blog entries from on how to Improve performance of the Spring-Petclinic application (Julien Dubois)
- Exception Handling in Spring MVC (Paul Chapman)
- Spring MVC Test Framework (Rossen Stoyanchev)
- Empower your CSS in your Maven build (Nicolas Frankel)